Final Project
Machine Learning (INFR 3700U)
Prepared by: Bhavik Naik (100617351), Khabeer Azizi (100699993)
Group 16
November 11, 2020

The Problem

In today's world, getting a university degree is quite challenging. There are financial, social, and mental health factors that are impacting students all across the world as they seek for a higher education. But, by far the most challenging part of a university degree is getting admitted. There are many standardized testing scores and other factors that students have to do well in order to be accepted into the university that they are applying to. In our dataset of 500 Indian students, we will try to predict if a student will be admitted into the university of choice. In this dataset, we have access to the students' GRE score, TOEFL score, their statement of purpose and letter of recommendation strength, the university rating that they are applying to, their CGPA, if they have research experience, and what the chance of being admitted is. We will be taking this data and feeding it into three different types of machine learning algorithms to predict the chance of admission. We hope to also feed the models a fictitious student and see if it will be able to calculate the percentage of admission. This will allow us to make a useful algorithm that will benefit students in the long term. They will be able to input their standard testing scores, as well as their averages, and see what chance of admission they have. We hope to pick the correct algorithms that will predict the admission chances correctly, and also be able to predict future students and their chances of getting admission.

Data Graphs

In this section, we will show 4 graphs that will show us a better representation of the data.

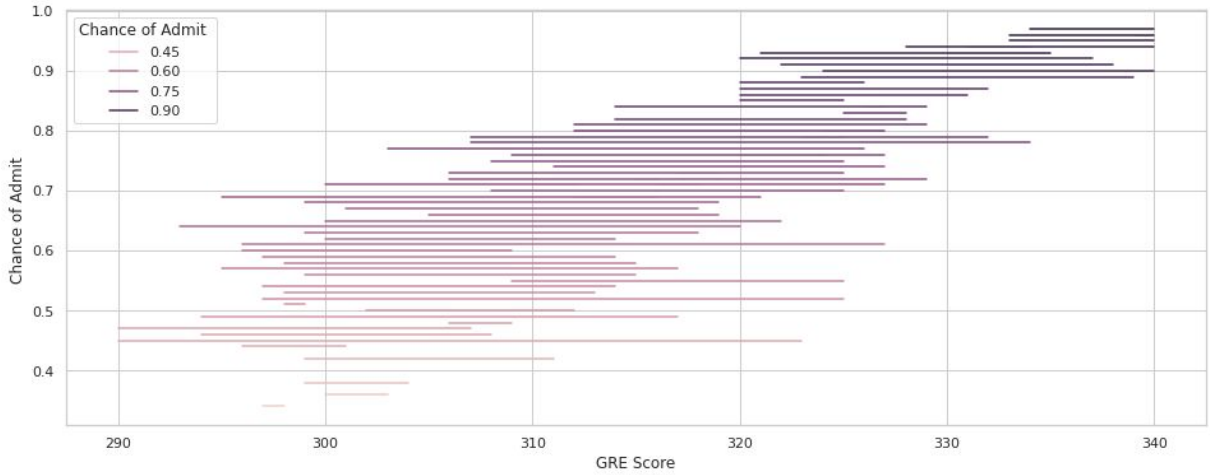

Figure 1 - The GRE score of the student vs the chance of admission

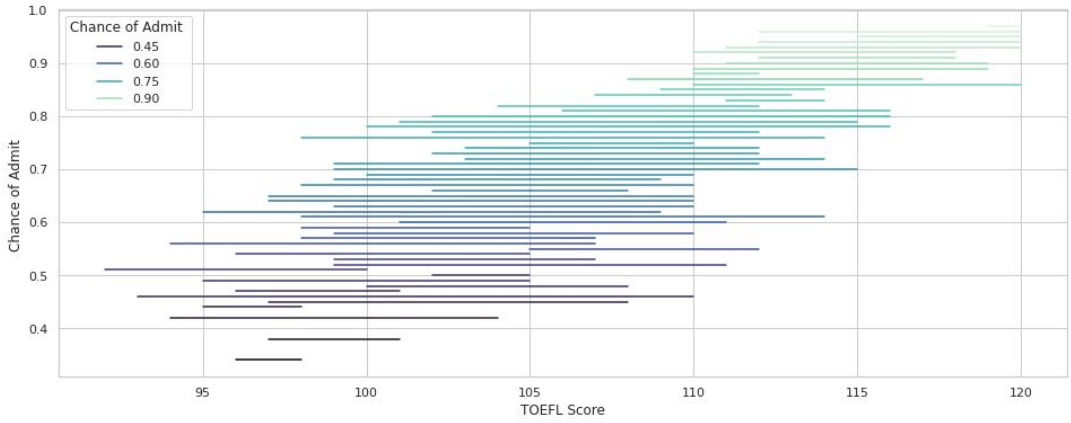

Figure 2 - The TOEFL scores of the students vs the chance of admission

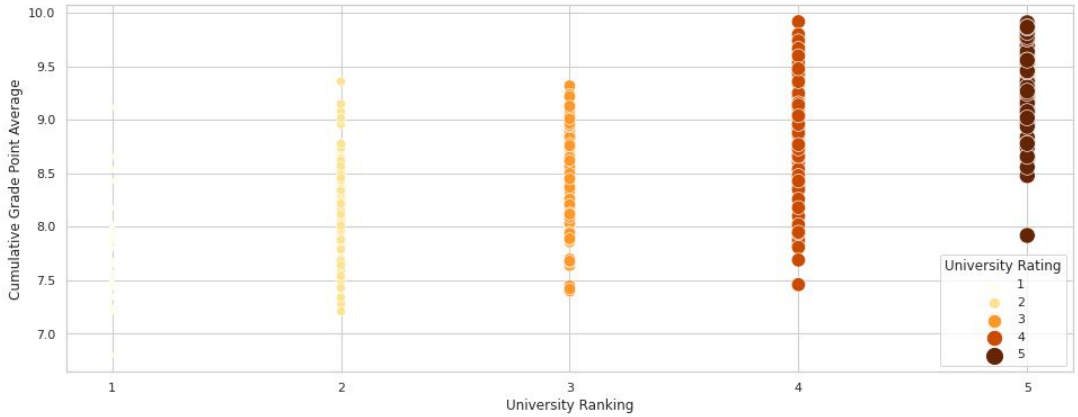

Figure 3 - The university rating vs the CGPA of the students

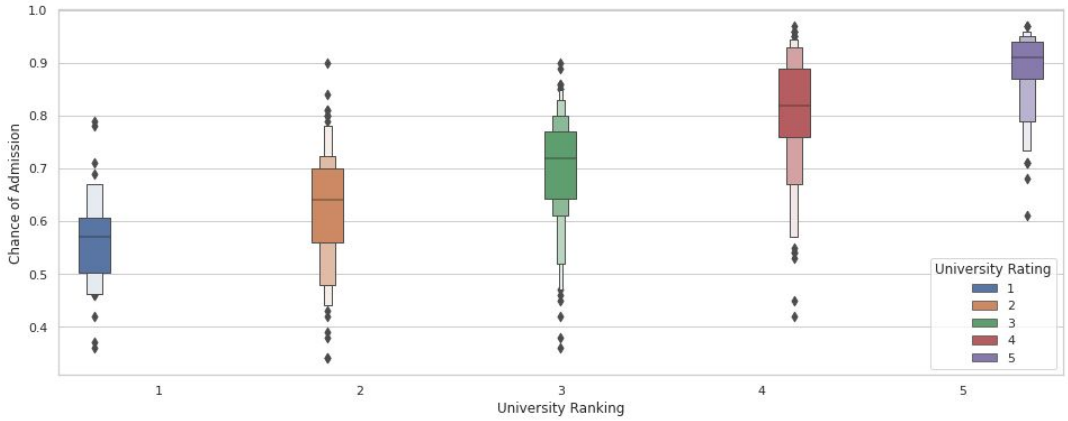

Figure 4 - The university rating vs the chance of admission

Preparation of Data

We will be preparing the data for the machine learning algorithms before running them. The first thing that we will do is drop the columns that we do not need, and those that will not have an impact on our machine learning algorithms. These columns include the serial no. of the students. This column had an index value of each student in the dataset. The second thing that we will be doing to clean up our dataset is dropping any blank values. When we ran the dropna() function, we realized that there were no blanks in our dataset. We then had to make sure that all

of the values that are listed in their respective columns were in fact all floats or integers. This check is important because we do not want to feed the machine learning algorithms inconsistent variable types. After this check was complete, we used the SkLearn train_test_split module to split the dataset into a training and testing dataset. We will be using a percentage of 50% to split the dataset. This will allow us to train out the models on half the dataset and then predict the testing dataset. We will be dropping the "Chance of Admit" column which contains the percentages of admission for each student. This is the value that we would like the models to predict. Now, our data is cleaned up, we can proceed in creating three machine learning algorithms to predict the chance of getting into university.

The Machine Learning Algorithms

We will be using 3 machine learning algorithms to predict the chance of admission. First, we will train the model with our data and figure out how reliable the algorithms are to predict the correct percentage from the test data. Then, we will use these algorithms to predict the chance of admission for two fictitious students. The first one will have a GRE score of 327, TOEFL score of 111, SOP of 4, LOR of 4, CGPA of 8.89, they have research experience, and they are applying to a university rating of 4. For the second student, they have a GRE score of 307, TOEFL score of 101, SOP of 3, LOR of 2.5, CGPA of 7.85, they have research experience, and they are applying to a university rating of 4. Furthermore, we will be evaluating all three models based on the mean squared error and the mean absolute error. Due to the fact that this is a regression problem, as we are going to be predicting a number, these two metrics will be a great way to see if our algorithms are in fact producing the correct results from the training and testing datasets. The lowest mean squared error and mean absolute error will be our best algorithm and we will later show graphs indicating the correctness of the algorithm.

First Algorithm - Deep Neural Network

The first algorithm that we will use is going to be a Deep Neural Network. This will be based on the Keras deep learning module. We will be using a sequential model from Keras to build our deep neural network and use the various optimization algorithms to tailor the algorithm to our data. As mentioned above, the loss will be set to the mean_squared_error and the secondary metric will be set as mean_absolute_error. In terms of the parameters that we used to build the neural network, we used 3 Dense layers, a Dropout layer, and a Flatten layer. The flatten layer came first and that flattened the input from all seven columns that we are feeding to the algorithm. This followed two Dense layers, both having 20 nodes each, and both having their activation set to "relu". The second last layer was the dropout layer, where we set the dropout rate to 20%. This means that 20% of the nodes will drop out when running. Finally, the last Dense layer, with a single node, came last to complete the deep neural network. In terms of the optimizer, we used the "rmsprop" optimizer to run the network. We set the epochs to 100 and let the training begin.

Results: Our best pass got a mean squared error of 0.0242 and a mean absolute error of 0.1139. This was for the training dataset. For the testing dataset, we got a mean squared error of 0.0115 and mean absolute error of 0.0863. These both have low numbers and we can see that the algorithm calculated the admission rate to a hundredths of a place. For the two fictitious students, the first student got a chance of admission of 0.715 and the second student got a chance of admission of 0.665.

Second Algorithm - Linear Regression

The second algorithm that we will use is the SkLearn Linear Regression model. This basic regression model is a great algorithm that will be able to predict our data quite accurately. As mentioned above, our problem is a regression problem, hence, we must use a regression model to represent the data. For our regression model, we have used the default parameters for the function and have trained the model on the training dataset and predicted the model on the testing dataset.

Results: The mean squared error for the model was 0.003325 and the mean absolute error was 0.041680 for the testing dataset. This model has predicted an accuracy to the thousandths of a place. This means that the linear regression model is better than the deep neural network. For the prediction of the two fictitious students, the model gave a percentage of admission of 0.812 for the first student, and 0.601 for the second student.

Third Algorithm - SGDRegressor

The third algorithm that we will use is the SGDRegressor. This algorithm is a regression algorithm that is tailored to a small number of rows of data. Unlike some of the larger regression models, this one needs to be scaled up and that is how the algorithm will output a viable prediction based on the data. We used the SkLearn StandardScaler function to scale up the testing data, and then, we called the SGDRegressor with all of its default values. After training it on the training dataset, we predicted the results of the testing dataset.

Results: For the mean squared error, the algorithm gave us a value of 0.023461, and for the mean absolute error, the algorithm gave us a value of 0.126593. This is significantly higher than both of the previous two algorithms, and we can see that this algorithm was not as efficient when compared to the previous machine learning algorithms. For the two fictitious students, it predicted the first student to have a 0.696 chance, and for the second student, the prediction was 0.709.

Comparative Performance Table

Algorithm	Mean Squared Error	Mean Absolute Error	Prediction for student 1	Prediction for student 2
Deep Neural Network	0.0115	0.0863	0.715	0.665
Linear Regression	0.003325	0.041680	0.811	0.601
SGDRegressor	0.023461	0.126593	0.696	0.709

Best Performing Algorithm

The best performing algorithm is the linear regression algorithm as it gave a mean squared error of 0.003325 and a mean absolute error of 0.041680. We can state this conclusion because we know that the lower both values are, the closer the algorithm was to predict the testing data. We will now showcase 4 graphs that showcase the algorithm and how well it performs when compared to the rest.

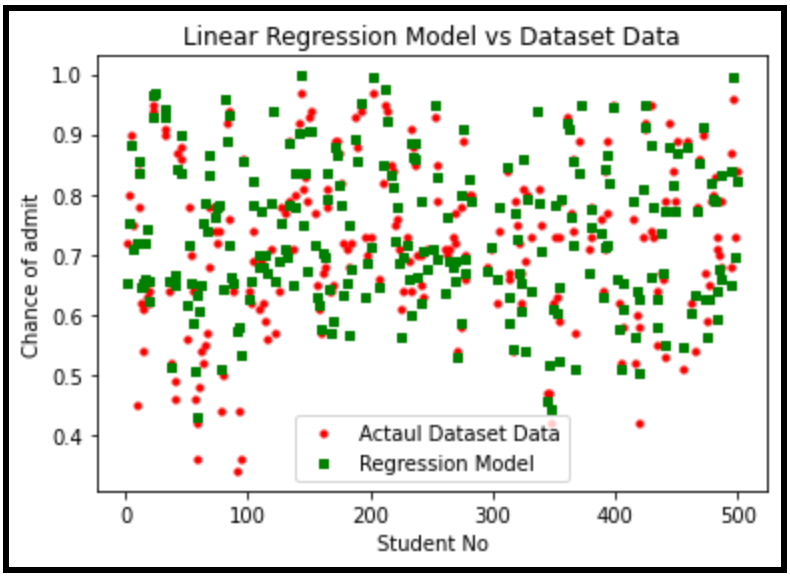

Figure 5: The student vs their chance of admission based on the actual data (red) and our model (green)

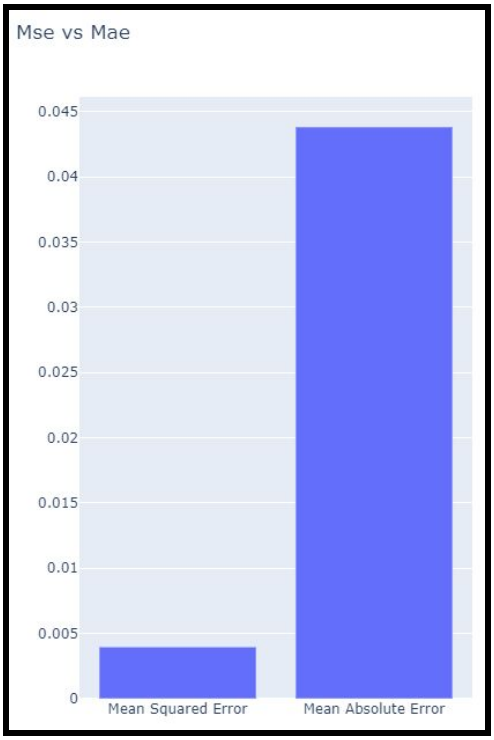

Figure 7: The two values for the mean squared error (MSE) and the mean absolute error (MAE) for the linear regression algorithm

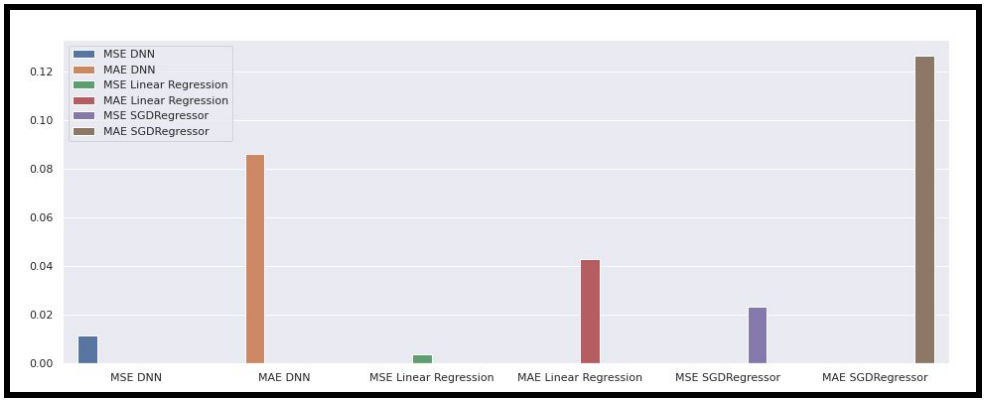

Figure 8: This graph shows the different MSE and MAE across all 3 of our algorithms

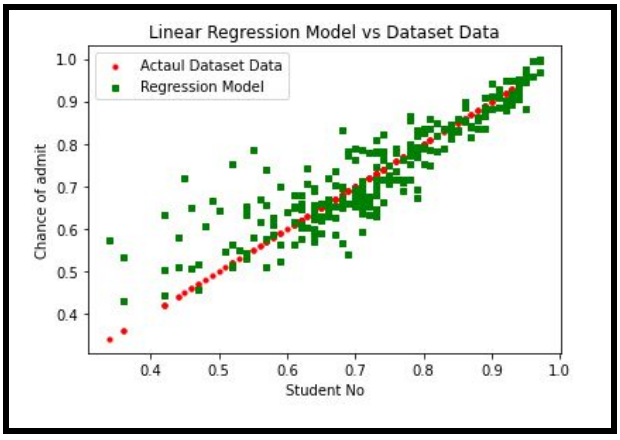

Figure 6: This is similar to figure 5 but here it is a regression plot. Which shows how as the model progressed the predictions got a lot closer to the actual data set.

Appendix

In the appendix, we will show all of the Python code that we used to achieve the results of the report.

```
# Import the various libaries

import pandas as pd
import os
import math
import numpy as np

import tensorflow as tf
from tensorflow import keras
import keras.optimizers
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

import plotly.graph_objects as go
import plotly.express as px

# Read in the CSV and look at the head
df=pd.read_csv('Admission_Predict_Ver1.1.csv')
df.head()

# Drop the Series No. column
df = df.drop(['Serial No.'], axis=1)

# Drop the blank values
df = df.dropna()

# create the training and testing data. Then create the 2 fictitious
students
from sklearn.model_selection import train_test_split
train, test = train_test_split(df, test_size=0.5)
train_labels = train.iloc[:,-1]
train_data = train.drop(['Chance of Admit '], axis=1)
test_labels = test.iloc[:,-1]
test_data = test.drop(['Chance of Admit '], axis=1)
train_labels_mc = train['Chance of Admit ']
```

```
test_labels_mc = test['Chance of Admit ']

fakeStudent1 = [[327, 111, 4, 4, 4, 8.89, 1]]
fakeStudent2 = [[307, 101, 4, 3, 2.5, 7.85, 1]]

# Create the Deep Neural Network and build the model with the nodes
model = keras.models.Sequential([
keras.layers.Flatten(input_shape=[7]),
keras.layers.Dense(20, activation="relu"),
keras.layers.Dense(20, activation="relu"),
keras.layers.Dropout(rate=0.2),
keras.layers.Dense(1)
])

model.summary()

# Compile the model with the correct loss, optimizers, and metrics
import keras.optimizers
model.compile(loss = "mean_squared_error",
optimizer="rmsprop",
metrics=["mean_absolute_error"])

# fit the training and testing datasets to the dnn model
history = model.fit(train_data, train_labels_mc, epochs=100,
validation_data=(test_data, test_labels_mc), verbose=1)# Turn
verbose=1 to printing epochs

#predict the chance of admission for the two fictitious students

dnnPred1 = model.predict(np.array(fakeStudent1))
print(dnnPred1)
dnnPred2 = model.predict(np.array(fakeStudent2))
print(dnnPred2)

# Build the linear regression algorithm and print out the MSE and
MAE. Fit the training data and predict the testing data

from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
```

```
reg = LinearRegression()
reg.fit(train_data, train_labels_mc)

admission = reg.predict(test_data)

from sklearn.metrics import mean_squared_error, mean_absolute_error
print(mean_squared_error(test_labels_mc, admission))
print(mean_absolute_error(test_labels_mc, admission))

#predict the chance of admission for the two fictitious students

lRPred1 = reg.predict(np.array(fakeStudent1))
print(lRPred1)
lRPred2 = reg.predict(np.array(fakeStudent2))
print(lRPred2)

#Build the SGDRegressor algorithm and scale it up. Also, train the
model using the training dataset and predict it with the testing
dataset. Print out the the MSE and MAE

from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import SGDRegressor

sgdreg = make_pipeline(StandardScaler(), SGDRegressor())
sgdreg.fit(test_data, train_labels_mc)
sgdadmission = sgdreg.predict(test_data)

print(mean_squared_error(test_labels_mc, sgdadmission))
print(mean_absolute_error(test_labels_mc, sgdadmission))

#predict the chance of admission for the two fictitious students

sgdPred1 = sgdreg.predict(np.array(fakeStudent1))
print(sgdPred1)
sgdPred2 = sgdreg.predict(np.array(fakeStudent2))
print(sgdPred2)
```

```
# Build the scatter plot showing the comparison between the linear
regression algorithm and the actual data

from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error

goodMse = mean_squared_error(test_labels_mc, admission)
goodmae = mean_absolute_error(test_labels_mc, admission)

goodMse = [goodmae]
goodmae = [goodmae]

fig = plt.figure()
ax1 = fig.add_subplot(111)
# Create the actual scatter plot
ax1.scatter(test_labels, test_labels_mc, s=10, c='red', marker="o",
label='Actaul Dataset Data')
ax1.scatter(test_labels, admission, s=10, c='green', marker="s",
label='Regression Model')
plt.legend(loc='upper left');
plt.xlabel('Student No')
plt.ylabel('Chance of admit')
plt.title('Linear Regression Model vs Dataset Data')
plt.legend()
plt.show()

# Create the bar graph showing both the MSE and MAE values for the
linear regression algorithm
goodMse = mean_squared_error(test_labels_mc, admission)
goodmae = mean_absolute_error(test_labels_mc, admission)
y = [goodMse, goodmae]
Value = ['Mean Squared Error','Mean Absolute Error']
fig = go.Figure([go.Bar(x=Value, y=y)])
fig.update_layout(height=700, width=500, title_text="Mse vs Mae")
fig.show()

# Create the bar graph showing the difference between the MSE and MAE
values across all three algorithms to see the various margins between
```

```
the algorithms

goodMse = mean_squared_error(test_labels_mc, admission)
goodmae = mean_absolute_error(test_labels_mc, admission)
# The various MSE and MAE from the algorithms
y = [0.0115, 0.0863, goodMse, goodmae, 0.023461, 0.126593]
colors = {'Deep Nerual Network': 'red',
          'Linear Regression': 'blue',
          'SGDRegressor': 'lightgreen',
          }
X = ['Deep Nerual Network', 'Linear Regression', 'SGDRegressor']
sns.set_theme(style="whitegrid")
sns.set(rc={'figure.figsize':(16,6)})
sns.barplot(data = df, x = Value , y = y, hue= Value)
fig.update_traces(marker_color='green', )
fig.show() # how do you change the color for every 2 bars?,
fig.update_traces(marker_color='green')

# Starting Graphs

#Import the various libraries
import pandas as pd
import os
import math
import numpy as np
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objects as go
import plotly.express as px
import seaborn as sns
from matplotlib import pyplot
import matplotlib as plt

# Create the horizontal line graph showing the GRE score vs the
chance of admission

un_rank = df['University Rating']

gre = df['GRE Score']
```

```
coa = df['Chance of Admit ']
sns.set_theme(style="whitegrid")
sns.lineplot(data=df, x="GRE Score", y="Chance of Admit ",
hue="Chance of Admit ")

# Create the horizontal line graph showing the TOEFL score vs the
chance of admission
sns.set_theme(style="whitegrid")
pallete = sns.color_palette("mako", as_cmap=True)

sns.lineplot(data=df, x="TOEFL Score", y="Chance of Admit ",
hue="Chance of Admit ", palette=pallete,)

# Create the scatter plot  showing the university ranking vs score vs
the CGPA

sns.set_theme(style="whitegrid")
pal = sns.color_palette("YlOrBr", as_cmap=True)
plt.xticks([1,2,3,4,5])
ax = sns.scatterplot(
    data=df, x='University Rating', y='CGPA', hue = 'University
Rating', size="University Rating", palette=pal,
    sizes=(20, 200), legend="full",
)
ax.set(xlabel="University Ranking", ylabel = "Cumulative Grade Point
Average")

# create the boxen plot that shows the university ranking and the
chance of admission
ur = df['University Rating']
ca = df['Chance of Admit ']

ax = sns.boxenplot(data = df, x = ur , y = ca, hue=ur)
ax.set(xlabel="University Ranking", ylabel = "Chance of Admission")
```